In the Claims

1. (Currently Amended) A method for maintaining data security comprising:

creating a package associated with a vault, the package comprising data bundled together with one or more permissions for regulating use of the data, the one or more permissions comprising one or more usage rule sets; and

providing a receiver for processing the package and storing the data in the vault,

the vault being dedicated hard drive space whose existence and contents are invisible to a user, wherein the existence and contents of the hard drive space are invisible to the user by an assignment of false file names and locations as seen by the user.

2. (Original) A method according to claim 1, wherein the step of processing the package further comprises opening the package and verifying the receiver for processing of the package.

3. (Original) A method according to claim 2 further comprising searching for at least one driver for reading the package.

4. (Original) A method according to claim 1 further comprising detecting violation of said one or more permissions.

5. (Original) A method according to claim 4 wherein the step of providing a receiver further comprises providing internal security.

6. (Original) A method according to claim 5, wherein the internal security comprises creating a tag file corresponding to the data and mapping the tag file against the data in a virtual table, with the virtual table including an actual file name of the data and a corresponding tag name for the tag file, wherein the virtual table and the data are stored in the vault.

7. (Original) A method according to claim 5, wherein the step of providing internal security comprises identifying an anchor address corresponding to an original location for at least one of the vault, a driver used for reading of the package and a database storing the permissions, combining the

addresses together to provide a key for regulating system operation and identifying when the key will not operate.

8. (Original) A method according to claim 5, wherein step of creating a package further comprises an executable for verifying the operation of the receiver when the package is opened.

9. (Original) A method according to claim 5, wherein the internal security comprises the monitoring of a registry comprising:

requesting a handle for a registry key to a calling process; requesting a registry key value for the handle; and obtaining security clearance to complete the requests.

10. (Original) The method of claim 9 further comprising after requesting a handle for a registry key to a calling process:

determining a process ID and registry key; determining whether the process is secured by checking a secured process list;

if the process is secured, determining whether the registry key is on a rejection list;

if the registry key is on the rejection list, denying the process access to the registry key; and

if the process is not on the secured list or if the registry key name is not on the rejection list, completing the request.

11. (Original) The method of claim 9 further comprising after requesting a registry key value for the handle:

determining a process ID and registry key value;

determining whether the process is secured by checking the secured process list;

if the process is secured, determining whether the registry key is on the rejection list;

if the registry key is on the rejection list, denying the process access to the registry key value;

if the process is not on the secured list, completing the request;

if the registry key is not on the rejection list and the process is on the secured process list, processing the value request and determining whether the value is on the rejection list;

if the value is not on the rejection list allowing the request to be completed; and

if the value is on the rejection list denying access to the registry key value.

12. (Original) The method of claim 9 further comprising after modifying and deleting handles and values:

determining a process ID;

determining whether the process is secured by checking whether the process is on the secured process list;

if the process is not on the secured process list, completing the request; and

if the process is on the secured process list, not allowing the request to be completed.

13. (Original) A method according to claim 5, wherein the step of providing internal security comprises a method for monitoring shared memory comprising:

providing a call to reserve a memory page for a requesting process;

filtering the reserve call according to whether the page can be shared;

providing a call to commit the memory page for the requesting process or for a subsequent requesting process; and

filtering the commit call according to whether the page can be shared and whether the process can be secured.

14. (Original) The method of claim 13 wherein filtering the reserve call comprises:

determining whether the page can be shared based on request parameters;

if the page cannot be shared, allowing the request to be completed; and

if the page can be shared, tracking the reserve call by creating a record and entering the record into a shared memory list.

15. (Original) The method of claim 14 wherein the record includes a process ID, page number and share count.

16. (Original) The method of claim 13 wherein filtering the commit call comprises:

determining, by accessing a shared memory list, if the page is shared by another process;

if the page is shared, determining whether either of the sharing processes are secured by accessing a secured process list;

if either process is secured, disallowing page sharing;

if both processes are not secured, creating a new shared memory record, updating the share count for processes sharing the page and updating the shared memory list with information contained in the new record; and

if the page is not shared, completing the commit request.

17. (Original) The method of claim 16 wherein the record includes a process ID, page number and share count.

18. (Original) The method of claim 13 further comprising:

providing a call to free the memory page of all address spaces;

determining whether the process is secured by checking a secured process list;

if the process is secured, overwriting the page to delete secured data, and deleting all records in the shared memory list with a page number the same as the overwritten page; and

if the process is not secured deleting all records from a shared memory list with a page number corresponding to the unsecured process page.

19. (Original) A method according to claim 5 wherein the step of providing internal security further comprises:

a vault provision step of providing a vault system for segregating vault data from other system data; and

a file system security driver provision step of providing a file system security driver which intercepts file system calls, and for each specific one of said intercepted file system calls, determines whether said specific one of said intercepted file system calls is from a process accessing said vault data, and, if said specific one of said intercepted file system calls is from a process accessing said vault data, permitting the file system call to create or modify data only within said vault system.

20. (Original) The method of claim 19, where said file system security driver provision step further comprises a file open handling step of, for each specific one of said intercepted file system calls which is a file open call, comprising the steps of:

determining whether said file open call is a request for data from among said vault data; and

if said file open call is a request for data from among said vault data, performing a check on process making said request to see if said process is already a secured process which has previously opened said data from among said vault data, and if so, allowing access to said vault data, and performing an access check on process making said request, and then processing the request by allowing access to said process which is not already a secured process if said access check is passed but not allowing access at all if said access check is not passed;

if said file open call is not a request for data from among said vault data, performing a check on said process making said request to see if said process is already a secured process, and passing the request onto an operating system if said process making said request is not a secured process, and, if said process making said request is a secured process, determining if file referred to in said file open call exists, and if it does, opening said file for read only, and if it does not, creating said file in said vault data.

21. (Original) The method of claim 20, wherein said processing the request by allowing access to said process which had not previously been granted access to said vault data comprises the step of:

querying user to determine if said user would like to open said data from among said vault data, and opening said data from among said vault data only if said user would like to open said data.

22. (Original) The method of claim 21, wherein said processing the request by allowing access to said process which had not previously been granted access to said vault data comprises the step of:

recording said allowed access and monitoring total accesses allowed.

23. (Original) The method of claim 20, wherein said processing the request by allowing access to said process which had not previously been granted access to said vault data comprises the step recording said process which had not previously been granted access to said vault data making said request in a list of processes allowed to access said vault data.

24. (Original) The method of claim 20, wherein said step of creating said file in said vault data comprises the step of :

sending said secured process a stand-in file handle; creating a corresponding vault file handle; and storing said stand-in file handle and said corresponding vault file handle.

25. (Original) The method of claim 20, wherein said step of opening said file for read only comprises the steps modifying any file request flags of said file open call which indicating modification of the file is permitted; and passing said modified file open call to said operating system.

26. (Original) The method of claim 19, wherein said file system security driver provision step further comprises a file request handling step of, for each specific one of said intercepted file system calls which is a file read/write call, comprising the steps of :

determining whether said read/write request is a request for data from among said vault data;

if said read/write request is a request for data from among said vault data, allowing access if process making said request is allowed to access said vault data; and

if said read/write request is a request for data not from among said vault data, allowing access if said process making said request is not allowed to access said vault data, and allowing access if said read/write request is a read request.

27. (Original) The method of claim 19, wherein said file system security driver provision step further comprises a file information request step, comprising the step of:

determining whether said file information request is a request regarding data from among said vault data, and if not, passing said file information request to said operating system, and if so, discerning correct file size and returning said correct file size.

28. (Original) The method of claim 19, wherein said file system security driver provision step further comprises a file change request step, comprising the step of:

determining whether said file change request is a request regarding data from among said vault data, and if so performing said file change request on said vault data, and if not, checking to see if the requesting process is a secured process, and if not, passing said file change request to said operating system, and if so, blocking the request.

29. (Original) The method of claim 19, wherein said file system security driver provision step further comprises:

a file open handling step of, for each specific one of said intercepted file system calls which is a file open call, comprising the steps determining whether said file open call is a request for data from among said vault data; and

if said file open call is a request for data from among said vault data, performing a check on process making said request to see if said process is already a secured process which has previously opened said data from among said vault data, and if so, allowing access to said vault data, and performing an access check on process making said request, and then processing the request by allowing access to said process which is not already a secured process if said access check is passed but not allowing access at all if said access check is not passed;

if said file open call is not a request for data from among said vault data, performing a check on said process making said request to see if said process is already a secured process, and passing the request onto an operating system if said process making said request is not a secured process, and, if said process making said request is a secured process, determining if file referred to in said file open call exists, and if it does, opening said file for read only, and if it does not, creating said file in said vault data;

a file read/write request handling step of, for each specific one of said intercepted file system calls which is a file read/write call, comprising the steps of :

determining whether said read/write request is a request for data from among said vault data;

if said request is a request for data from among said vault data, allowing access if process making said request is allowed to access said vault data; and

if said request is a request for data not from among said vault data, allowing access if said process making said request is not allowed to access said vault data, and allowing access if said read/write request is a read request;

a file information request step, comprising the step of :

determining whether said file information request is a request regarding data from among said vault data, and if not, passing said file information request to said operating system, and if so, discerning correct file size and returning said correct file size;

and

a file change request step, comprising the steps determining whether said file change request is a request regarding data from among said vault data, and if so performing said file change request on said vault data, and if not, checking to see if the requesting process is a secured process, and if not, passing said file change request to said operating system, and if so, blocking the request.

30. (Original) A method according to claim 5, wherein the step of providing internal security further comprises monitoring a system clock of a computer to prevent unauthorized access to data comprising the steps of:

initializing a clock monitor comprising the steps of:

reading a first time value from the system clock; determining whether a permissions database having one or more clock-related permission field each field comprising one or more clock-related permissions, and a stored time value field comprising a stored time value, is initialized on the computer system;

if the permissions database is initialized, comparing the first time value to the stored time value and, if the first time value is later than the stored time value, storing the first time value in the stored time value field, if the first time value is earlier than the stored time value, disabling the one or more clock-related permissions, whereby disabling the clock-related permissions prevents access to the data; and

if the permissions database is not initialized, storing the first time value in the stored time value field.

31. (Original) The method of claim 30 wherein the step of determining whether the permissions database is initialized comprises the step of:

reading the stored time value from the stored time value field in the permissions database, and if the stored value is zero, concluding that the permissions database is not initialized, and if the stored time value field is other than zero, concluding that the permissions database is initialized.

32. (Original) The method of claim 30 further comprising the steps of :

tracking a true system time, which is the stored time value plus an internal elapsed time measured from initialization of the clock monitor;

after a predetermined tracking interval, reading a second time value from the system clock;

comparing the second time value with the true system time and generating a time deviation based on the comparison;

if the time deviation is not within an acceptable deviation, disabling the one or more clock-related permissions;

if the time deviation is within the acceptable deviation, enforcing the clock-related permissions ; and

storing the true system time.

33. (Original) The method of claim 32, after the step of if the time deviation is not within an acceptable deviation, disabling one or more clock-related permissions, further comprising the steps of:

reading a third time value from the system clock;

comparing the third time value with the internal elapsed time;

generating a second time deviation based on the comparison; and

if the second time deviation is within the acceptable deviation, reenabling the clock-related permissions, storing the true system time in the stored time value field, and storing the third time value a last known good system time value field in the permissions database.

34. (Original) The method of claim 32 wherein the predetermined tracking interval is substantially in the range of zero seconds to sixty seconds.

35. (Original) The method of claim 32 wherein the accepted deviation is substantially in the range of zero seconds to three hours.

36. (Original) The method of claim 30 further comprising the steps of:

tracking a true system time, which is the stored time value plus a true system time measured from initialization of the clock monitor;

reading a second time value from the system clock;

comparing the second time value with the true system time and generating a time deviation based on the comparison; and

if the time deviation is within an acceptable deviation, storing the second time value in the stored time value field.

37. (Original) The method of claim 36 further comprising the step of powering down the computer.

38. (Original) The method of claim 36 wherein the accepted deviation is substantially in the range of zero seconds to three hours.

39. (Original) The method of claim 30 wherein the clock-related permissions comprise date-related permissions.

40. (Original) A method according to claim 5 for providing data security in a first device driver operably installed in a computer operating system having a layered plurality of device drivers for accessing data in a data storage device, the method comprising the steps of :

detecting an I/O request to said first device driver;

determining whether said first device driver is functionally uppermost in the layered plurality of device drivers;

if said first device driver is functionally uppermost in the layered plurality of device drivers, performing the I/O request in said first device driver; and

if said first device driver is not functionally uppermost in the layered plurality of device drivers, denying the I/O request in said first device driver, and allowing the I/O request to be performed by a next lower-level device driver in the layered plurality of device drivers.

41. (Original) The method of claim 40 wherein said first device driver is a file system monitor.

42. (Original) The method of claim 40 wherein the data is stored in a secure virtual file system, and wherein the step of performing the I/O request further comprises the step of implementing data security measures.

43. (Original) The method of claim 40 wherein the data is stored in encrypted form, and wherein the step of performing the I/O request further comprises the step of decrypting the data.

44. (Original) The method of claim 40 wherein the step of performing the I/O request further comprises the step of checking the data for viruses.

45. (Original) The method of claim 40 wherein the step of determining whether said first device driver is functionally uppermost in the layered plurality of device drivers further comprises the steps of:

determining whether said first device driver has been previously called;

if said first device driver has not been previously called, detecting an initial calling module address, storing said initial calling module address, and concluding that said first device driver is functionally uppermost in the layered plurality of device drivers;

if said first device driver has been previously called, detecting a second calling module address, comparing said second calling module address to the initial calling module address, and concluding that said first device driver is functionally uppermost in the layered plurality of device drivers only if the initial calling module address matches the second calling module address.

46. (Original) The method of claim 40 wherein the step of denying the I/O request in the secure first device driver comprises the steps of:

setting a first device driver shutdown flag; and

initiating a re-hook process.

47. (Original) The method of claim 40 further comprising, after the step of detecting an I/O request to said first device driver, the steps of:

checking whether a first device driver shutdown flag is set; and

if said first device driver shutdown flag is set, omitting further steps in said first device driver, and allowing the I/O request to be performed by a next lower-level device driver in the layered plurality of device drivers.

48. (Original) The method of claim 47 wherein the step of initiating a re-hook process further comprises the steps of:

counting the number of times the re-hook process has been initiated;

checking whether the number of times has reached a predetermined maximum threshold;

if the number of times has reached a predetermined maximum threshold, initiating a programmable security response;

if the number of times has not reached a predetermined maximum threshold, initiating re-attachment of said first device driver functionally uppermost in the layered plurality of device drivers;

if said first device driver has been reattached functionally uppermost in the layered plurality of device drivers, unsetting said first device driver shutdown flag; and concluding the re-hook process.

49. (Original) The method of claim 48 wherein the programmable security response comprises the step of destroying the data.

50. (Original) The method of claim 48 wherein the data is stored in a secure virtual file system, and wherein the step of destroying the data further comprises the step of destroying the secure virtual file system.

51. (Original) The method of claim 48 wherein the programmable security response comprises the step of terminating open applications.

52. (Original) The method of claim 48 wherein the programmable security response comprises the step of destroying said first device driver on the data storage device.

53. (Original) The method of claim 48 wherein the programmable security response comprises the step of halting the operation of the computer.

54. (Original) The method of claim 48 wherein the programmable security response comprises the step of causing the computer to enter a state requiring reboot.

55. (Original) A method according to claim 1 further comprising:

a port request detection step of detecting a port request for use of a port sent by a process;

a process identification step of determining the identity of said requesting process;

a process check step of determining if said process should be permitted to access said port; and

a permit/deny step of allowing said port request to be fulfilled if said process should be permitted to access said port and denying said port request if said process should not be permitted to access said port.

56. (Original) The method of claim 55 wherein said process check step comprises: a secure process list check step of determining whether said process appears on a list of secure processes.

57. (Original) The method of claim 55 further comprising:
a tracking step of tracking said port request.

58. (Original) A method according to claim 5, wherein the step of providing internal security further comprises:

a port request detection step of detecting a port request for use of a port sent by a process;

an open port process identification step of, if said port request is an open port request, determining the identity of said requesting process;

an open port process check step of, if said port request is an open port request, determining if said process should be permitted to open said port;

an open port permit/deny step of, if said port request is an open port request, allowing said open port request to be fulfilled and tracking said open port request if said process should be permitted to open said port and denying said port request if said process should not be permitted to open said port;

a close port process completion step of, if said port request is a close port request, completing said port request; and

a close port logging step of logging the closing of said port.

59. (Original) The method according to claim 58 where said open port process check step comprises:

a secure process list check step of determining whether said process appears on a list of secure processes.

60. (Original) The method according to claim 58 where said tracking of said open port request comprises keeping a log of process ID and returned port handle for said open port request, and said close port logging step of tracking the closing of said port comprises removing from said log said record of process ID and returned port handle for that port close request.

61. (Original) The method according to claim 60 further comprising:

a security check step comprising the steps of checking whether a process has open ports, and denying security clearance for a process with open ports, and allowing security clearance for a process with no open ports.

62. (Original) The method according to claim 61 wherein said open port process check step of comprises determining if said process identity appears on a secured process list, and where said step

of allowing security clearance for a process with no open ports comprises the step of placing said process on said secured process list.

63. (Original) A method according to claim 5, wherein the step of providing internal security further comprises:

a network request detection step of detecting a network request for use of a network sent by a process;

a process identification step of determining the identity of said requesting process;

a process check step of determining if said process should be permitted to access said network; and

a step of allowing said network request to be fulfilled if said process should be permitted to access said network and denying said network request if said process should not be permitted to access said network.

64. (Original) The method according to claim 63 wherein said process check step comprises: a secure process list check step of determining whether said process appears on a list of secure processes.

65. (Original) The method according to claim 64, wherein said network requests interface is the Transport Data Interface.

66. (Original) A method according to claim 1, wherein the step of creating a package comprises:

receiving a file of data for packaging; receiving a permissions database having one or more permissions associated with the file of data, the one or more permissions governing a client's use of the file;

generating a package global unique identifier; generating a package of data comprising the file, the one or more permissions and the global unique identifier; encrypting the package; and

generating a computer executable file comprising the encrypted package.

67. (Original) The method of claim 66 wherein the one or more permissions are selected from the group consisting an access count permission, an access time permission, an expiration date permission, an authorization date permission, a clipboard permission, a print permission, an unlimited access permission, an application permission, and a system-events permission.

68. (Original) The method of claim 67 further comprising the step of setting a password for access to the computer executable file.

69. (Original) The method of claim 68 wherein the package of data further comprises a recipient global unique identifier and further comprising the step of receiving the recipient global unique identifier after the step of generating a package global unique identifier.

70. (Original) The method of claim 69 wherein the package of data further comprises a client software.

71. (Original) A method according to claim 1 further comprising:

receiving a file of data for packaging;

receiving a package permissions database having one or more permissions associated with the file of data, the one or more permissions governing a client's use of the file;

generating a package global unique identifier;

generating a package of data comprising the file of data, the one or more permissions, the global unique identifier, and a client software;

encrypting the package;

generating a computer executable file comprising the encrypted package;

receiving the computer executable file at a client computer system having an operating system;

executing the computer executable file at the client computer system comprising the steps determining whether the operating system is a compatible operating system, and if so, executing a client software on the client computer system, the execution of the client software creating a client permissions database and a vault on the client computer system; and

determining whether the encrypted package is valid, and if so, recording the package global unique identifier in the client permissions database, extracting the file of data and the one or more permissions from the package of data, storing the file of data in the vault and storing the one or more permissions in the client permissions database, and if not, setting a state in the computer executable file to indicate that the package is installed.

72. (Original) The method of claim 71 further comprising the step of determining whether a second package is loaded on the computer system, and if so, terminating the second package, before the step of executing a client software on the client computer system.

73. (Original) The method of claim 72 wherein the step of determining whether the package is valid comprises the steps of searching the client permissions database for the package global unique identifier and, concluding that the package is valid if the package global unique identifier is not in the client permissions database, and concluding that the package is invalid if the package global unique identifier is not in the client permissions database.

74. (Original) The method of claim 73 wherein the package further comprises the client software having a version designation and, before the step of executing the client software, determining whether a second version of the client software is installed on the client computer system, and if not, extracting the client software from the package and installing the client software on the client computer system.

75. (Original) The method of claim 74 wherein if a second version of the client software is installed on the client computer system, determining whether the version designation of the client software installed on the client computer system is earlier than the second version, and if so, extracting the client software from the package and installing the client software on the client computer system.

76. (Original) The method of claim 74 wherein the client software comprises one or more device drivers and the client permissions database and the vault are generated by at least one of the one or more device driver.

77. (Original) The method of claim 71 wherein the client software comprises one or more device drivers and the client permissions database and the vault are generated by at least one of the one or more device driver.

78. (Original) The method of claim 71 wherein the package further comprises a receiver global unique identifier, and wherein the step of determining whether the encrypted package is valid comprises the steps of searching the client permissions database for a second receiver global unique identifier, and if not found, concluding that the package is invalid, and if found, comparing the receiver global unique identifier to the second receiver global unique identifier, determining whether they match, and if so, concluding that the package is valid, and if not, concluding that the package is invalid.

79. (Original) The method of claim 71 wherein the one or more permissions are selected from the group consisting an access count permission, an access time permission, an expiration date permission, an authorization date permission, a clipboard permission, a print permission, an unlimited access permission, an application permission, and a system-events permission.

80. (Original) The method of claim 71 wherein the computer executable file is password protected.

81. (Original) A method according to claim 5, wherein the step of providing internal security comprises:

detecting a file system request; completing said file system request;

receiving return information from said file system request;

determining whether said file system request is for a tag file associated with a secured file; and

if so, modifying said return information to reflect a file attribute of the secured file.

82. (Original) The method of claim 81 wherein said file attribute is file size.

83. (Original) The method of claim 81 wherein the step of determining further comprises the steps of :

determining whether said return information identifies a plurality of tag files associated with a plurality of secured files; and

if so, modifying said return information to reflect a file attribute of the plurality of secured files.

84. (Original) The method of claim 81 wherein the secured file is stored in encrypted form.

85. (Original) The method of claim 81 wherein the secured file is stored in a secure virtual file system.

86. (Original) The method of claim 81 wherein the secured file is stored on a remote networked device.

87. (Original) The method of claim 81 wherein the file system request is to open a file.

88. (Original) The method of claim 81 wherein the file system request is to delete a file.

89. (Original) The method of claim 81 wherein the file system request is to rename a file.

90. (Original) The method of claim 81 wherein the file system request is to query file information.

91. (Original) The method of claim 83 wherein the file system request is to set file information.

92. (Original) The method of claim 83 wherein the file system request is to find a first matching file.

93 (Original) The method of claim 83 wherein the file system request is to find a next matching file.

94. (Original) The method of claim 83 wherein the file system request is directory control.

95. (Currently Amended) A system for maintaining data security comprising:

a receiver for processing a package associated with a vault, the package comprising data bundled together with one or more permissions for regulating use of the data, the one or more permissions comprising one or more usage rule sets; and

the vault located within the receiver for storing the data, the vault being dedicated hard drive space whose existence and contents are invisible to a user, wherein the existence and contents of the hard drive space are invisible to the user by an assignment of false file names and locations as seen by the user.

96. (Original) A system according to claim 95 further comprising internal security for protecting the data stored in the vault.

97. (Original) A system according to claim 96, wherein the internal security further detects violation of said one or more permissions.

98. (Original) A system according to claim 97, wherein the internal security comprises a tag file corresponding to the data, and a virtual table mapping the tag file against the data by using an actual file name for the data and a tag name for the tag file, wherein the virtual table and data are stored in the vault.

99. (Original) A system according to claim 97, wherein the internal security comprises an anchor address corresponding to an original location for at least one of the vault, a driver used for reading of the package and a database storing the permissions, combining the addresses together to provide a key for regulating system operation and identifying when the key will not operate.

100. (Original) A system according to 97, wherein the internal security further comprises a registry monitoring system comprising:

a handle for a registry key to a calling process;

a registry key value for the handle;

a process ID and registry key;

security clearance to complete the requests;

wherein the process is secured by checking a secured process list;

if the process is secured, determining whether the registry key is on a rejection list;

if the registry key is on the rejection list, denying the process access to the registry key; and

if the process is not on the secured list or if the registry key name is not on the rejection list, completing the request.

101. (Original) A system according to claim 97, wherein the internal security comprises a shared memory system comprising:

a call to reserve a memory page for a requesting process;

the reserve call filtered according to whether the page can be shared;

a call to commit the memory page for the requesting process or for a subsequent process;

the commit call filtered according to whether the page can be shared and whether the process can be secured.

102. (Original) A system according to claim 97 wherein the internal security further comprises:

a vault system for segregating vault data from other system data; and

a file system security driver which intercepts file system calls, and for each specific one of said intercepted file system calls, determining whether said specific one of said intercepted file system calls is from a process accessing said vault data, and, if said specific one of said intercepted file system calls is from a process accessing said vault data, permitting the file system call to create or modify data only within said vault system.

103. (Original) A system according to claim 97, wherein the internal security further comprises a system for monitoring a system clock of a computer to prevent unauthorized access to data comprising:

reading a first time value from the system clock;

determining whether a permissions database having one or more clock-related permission field each field comprising one or more clock-related permissions, and a stored time value field comprising:

a stored time value, is initialized on the computer system;

if the permissions database is initialized, comparing the first time value to the stored time value and, if the first time value is later than the stored time value, storing the first time value in the stored time value field, if the first time value is earlier than the stored time value, disabling the one or more clock-related permissions, whereby disabling the clock-related permissions prevents access to the data; and

if the permissions database is not initialized, storing the first time value in the stored time value field.

104. (Original) A system according to claim 97, wherein the internal security comprises:

detecting an I/O request to said first device driver;

determining whether said first device driver is functionally uppermost in the layered plurality of device drivers;

if said first device driver is functionally uppermost in the layered plurality of device drivers, performing the I/O request in said first device driver; and

if said first device driver is not functionally uppermost in the layered plurality of device drivers, denying the I/O request in said first device driver, and allowing the I/O request to be performed by a next lower-level device driver in the layered plurality of device drivers.

105. (Original) A system according to claim 97, wherein the internal security comprises:

a port request detection step of detecting a port request for use of a port sent by a process;

a process identification step of determining the identity of said requesting process;

a process check step of determining if said process should be permitted to access said port; and

a step of allowing said port request to be fulfilled if said process should be permitted to access said port and denying said port request if said process should not be permitted to access said port.

106. (Original) A system according to claim 97, wherein internal security comprises:

a port request detection step of detecting a port request for use of a port sent by a process;

an open port process identification step of, if said port request is an open port request, determining the identity of said requesting process;

an open port process check step of, if said port request is an open port request, determining if said process should be permitted to open said port;

an open port step of, if said port request is an open port request, allowing said open port request to be fulfilled and tracking said open port request if said process should be permitted to open said port and denying said port request if said process should not be permitted to open said port;

a close port process completion step of, if said port request is a close port request, completing said port request; and a

close port logging step of logging the closing of said port.

107. (Original) A system according to claim 97, wherein internal security comprises:

a network request detection step of detecting a network request for use of a network sent by a process;

a process identification step of determining the identity of said requesting process;

a process check step of determining if said process should be permitted to access said network; and a

step of allowing said network request to be fulfilled if said process should be permitted to access said network and denying said network request if said process should not be permitted to access said network.

108. (Original) A system according to claim 96 comprising:

a machine readable medium having information packaging software that generates a computer executable file comprising a package of information, the package of information comprising:

a file of data; a permissions database having one or more permissions associated with the file of data;

an encryption software;

a network in communication with the machine readable medium;

a client computer system in communication with the network, the computer system adapted to receive the package of information and execute the computer executable file, the computer system having a client permissions database and a vault adapted to receive the package of information.

109. (Original) The system of claim 108 wherein the package of information further comprises a package global unique identifier, and the client computer system further comprises a module of computer code adapted to read the package global unique identifier, search the client permissions database for the package global unique identifier, and reject the package if the package global unique identifier is found in the client permissions database.

110. (Original) The system of claim 109 wherein the package of information further comprises a recipient global unique identifier, and the client computer system further comprises a module of computer code adapted to read the recipient global unique identifier, search the client permissions database for the recipient global unique identifier, and reject the package if the recipient global unique identifier is not found in the client permissions database.

111. (Original) The system of claim 110 wherein the one or more permissions are selected from the group consisting an access count permission, an access time permission, an expiration date permission, an authorization date permission, a clipboard permission, a print permission, an unlimited access permission, an application permission, and a system-events permission.

112. (Original) A system according to claim 97, wherein the system comprises a device driver for accessing data, the device driver operably installed in an operating system on an electronic computer, wherein said device driver:

detects a file system request; completes said file system request;

receives return information from said file system request;

determines whether said file system request is for a tag file associated with a secured file; and if so, modifies said return information to reflect a file attribute of the secured file.

113. (Original) The system of claim 112 wherein said file attribute is file size.

114. (Original) The system of claim 113 wherein said device driver further determines whether said return information identifies a plurality of tag files associated with a plurality of secured files; and if so, modifies said return information to reflect a file attribute of the plurality of secured files.

115. (Original) The system of claim 114 wherein said first device driver is a file system monitor.

116. (Original) The system of claim 114 wherein the secured file is stored in encrypted form.

117. (Original) The system of claim 114 wherein the secured file is stored in a secure virtual file system.

118. (Original) The system of claim 114 wherein the secured file is stored on a remote networked device.

119. (Original) The system of claim 114 wherein the file system request is to open a file.

120. (Original) The system of claim 114 wherein the file system request is to delete a file.

121. (Original) The system of claim 114 wherein the file system request is to rename a file.

122. (Original) The system of claim 114 wherein the file system request is to query file information.

123. (Original) The system of claim 114 wherein the file system request is to set file information.

124. (Original) The system of claim 123 wherein the file system request is to find a first matching file.

125. (Original) The system of claim 123 wherein the file system request is to find a next matching file.

126. (Original) The system of claim 123 wherein the file system request is directory control.

127. (Original) A system according to 97, further comprising a port blocking system, wherein said port blocking system operates to detect a port request for use of a port sent by a process; determine the identity of said requesting process; determine if said process should be permitted to access said port; and allow said port request to be fulfilled if said process should be permitted to access said port and deny said port request if said process should not be permitted to access said port.

128. (Original) A system according to 97, further comprising a network blocking system, wherein said network blocking system operates to determine the identity of said requesting process; determine if said process should be permitted to access said network; and allow said network request to be fulfilled if said process should be permitted to access said network and deny said network request if said process should not be permitted to access said network.

129. (Currently Amended) A computer program product for monitoring data security embodied in a memory medium that when read out directs a system to perform at least one of:

creating a package associated with a vault, the package comprising data bundled together with one or more permissions for regulating use of the data, the one or more permissions comprising one or more usage rule sets; and

opening the package and storing the data in the vault for restricted access of the data, the vault being dedicated hard drive space whose existence and contents are invisible to a user, wherein the existence and contents of the hard drive space are invisible to the user by an assignment of false file names and locations as seen by the user.

130. (Previously Presented) A computer program product according to claim 129 further directing the system to detect violations of said one or more permissions.

131. (Previously Presented) A computer program product according to claim 130 further directing the system to create a tag file corresponding to the data in the vault and to map the tag file against the data in a virtual table, with the virtual table stored in the vault and including an actual file name of the data and a corresponding tag name for the tag file.

132. (Previously Presented) A computer program product according to claim 130, further directing the system to provide internal security including identifying an anchor address corresponding to an original location for at least one of the vault, a driver used for reading of the package and a database storing the permissions, combining the addresses together to provide a key for regulating system operation and identifying when the key will not operate.

133. (Previously Presented) A computer program product according to claim 130 further directing the system to monitor a registry via:

requesting a handle for a registry key to a calling process;

requesting a registry key value for the handle; and

obtaining security clearance to complete the requests.

134. (Previously Presented) A computer program product according to claim 130 further directing the system to monitor shared memory via:

providing a call to reserve a memory page for a requesting process;

filtering the reserve call according to whether the page can be shared;

providing a call to commit the memory page for the requesting process or for a subsequent requesting process; and

filtering the commit call according to whether the page can be shared and whether the process can be secured.

135. (Previously Presented) A computer program product according to claim 130 further directing the system to:

provide a vault system for segregating vault data from other system data; and

provide a file system security driver which intercepts file system calls, and for each specific one of said intercepted file system calls, determines whether said specific one of said intercepted file system calls is from a process accessing said vault data, and, if said specific one of said intercepted file system calls is from a process accessing said vault data, permitting the file system call to create or modify data only within said vault system.

136. (Previously Presented) A computer program product according to claim 130, further directing the system to monitor a system clock of a computer to prevent unauthorized access to data by:

initializing a clock monitor via:

reading a first time value from the system clock;

determining whether a permissions database having one or more clock-related permission fields, each field comprising one or more clock-related permissions, and having a stored time value field comprising a stored time value, is initialized on the system;

if the permissions database is initialized, comparing the first time value to the stored time value and, if the first time value is later than the stored time value, storing the first time value in the stored time value field, if the first time value is earlier than the stored time value, disabling the one or more clock-related permissions, whereby disabling the clock-related permissions prevents access to the data; and

if the permissions database is not initialized, storing the first time value in the stored time value field.

137. (Previously Presented) A computer program product according to claim 130 further directing the system to:

detect an I/O request to a first device driver;

determine whether said first device driver is functionally uppermost in a layered plurality of device drivers;

if said first device driver is functionally uppermost in the layered plurality of device drivers, perform the I/O request in said first device driver; and

if said first device driver is not functionally uppermost in the layered plurality of device drivers, deny the I/O request in said first device driver, and allow the I/O request to be performed by a next lower-level device driver in the layered plurality of device drivers.

138. (Previously Presented) A computer program product according to claim 129 further directing the system to:

detect a port request for use of a port sent by a process;

determine the identity of said requesting process;

determine if said requesting process should be permitted to access said port; and

allow the port request if said process should be permitted to access said port and deny the port request if said process should not be permitted to access said port.

139. (Previously Presented) A computer program product according to claim 130 further directing the system to:

detect a port request for use of a port sent by a process;

if said port request is an open port request, determine the identity of said requesting process;

if said port request is an open port request, determine if said process should be permitted to open said port;

if said port request is an open port request, allow the open port request and track said open port request if said process should be permitted to open said port and deny said port request if said process should not be permitted to open said port;

if said port request is a close port request, complete said port request; and

log the closing of said port.

140. (Previously Presented) A computer program product according to claim 130 further directing the system to:

detect a network request for use of a network sent by a process;

determine an identity of said requesting process;

determine if said process should be permitted to access said network; and

allow the network request if said process should be permitted to access said network and deny the network request if said process should not be permitted to access said network.

141. (Previously Presented) A computer program product according to claim 130 comprising a package of information comprising:

a file of data;

a permissions database having one or more permissions associated with the file of data, the one or more permissions governing a client's use of the file;

a package global unique identifier; and a receiver global unique identifier.

142. (Previously Presented) The computer program product of claim 141 wherein the one or more permissions are selected from the group consisting of an access count permission, an access time permission, an expiration date permission, an authorization date permission, a clipboard permission, a print permission, an unlimited access permission, an application permission, and a system-events permission.

143. (Previously Presented) The computer program product of claim 142 further comprising a client software.

144. (Previously Presented) A computer program product according to 130 further comprising a device driver program for directing the system to access data, said device driver program comprising instructions for directing the system to:

detect a file system request;

complete said file system request;

receive return information from said file system request;

determine whether said file system request is for a tag file associated with a secured file; and

modify said return information to reflect a file attribute of the secured file, if said file system request is for a tag file associated with a secured file.

145. (Previously Presented) The computer program product of claim 144 wherein the device driver program further comprises instructions for directing the system to:

determine whether said return information identifies a plurality of tag files associated with a plurality of secured files; and

modify said return information to reflect a file attribute of the plurality of secured files, if said return information identifies a plurality of tag files associated with a plurality of secured files.

146. (Previously Presented) A computer program product according to claim 130 further directing the system to:

protect secure data by implementing a port blocking system which operates to detect a port request for use of a port sent by a process;

determine an identity of said requesting process;

determine if said process should be permitted to access said port; and allow said port request to be fulfilled if said process should be permitted to access said port and deny said port request if said process should not be permitted to access said port.

147. (Previously Presented) A computer program product according to claim 130 further directing the system to:

protect secure data by implementing a network blocking system which operates to determine an identity of said requesting process;

determine if said process should be permitted to access said network; and

allow said network request to be fulfilled if said process should be permitted to access said network and deny said network request if said process should not be permitted to access said network.

148. (Currently Amended) A system for maintaining security during transmission of data between at least two computers comprising:

a first computer having a system for creating a package associated with a vault, the package comprising data bundled together with one or more permissions selected from a list of available permissions for regulating use of the data, the one or more permissions comprising one or more usage rule sets; and

a second computer having a system for receiving the package from the first computer, opening the package upon verification and storing the data in the vault, the vault being dedicated hard drive space whose existence and contents are invisible to a user, wherein the existence and contents of the hard drive space are invisible to the user by an assignment of false file names and locations as seen by the user.

149. (Original) A system according to 148 further comprising internal security, wherein the internal security comprises a plurality of:

detecting violation of said one or more permissions;

creating a tag file corresponding to the data and mapping the tag file against the data in a virtual table, with the virtual table including an actual file name of the data and a corresponding tag name for the tag file, wherein the virtual table and the data are stored in the vault;

identifying an anchor address corresponding to an original location for at least one of the vault, a driver used for reading of the package and a database storing the permissions, combining the addresses together to provide a key for regulating system operation and identifying when the key will not operate; monitoring a registry comprising:

requesting a handle for a registry key to a calling process;

requesting a registry key value for the handle; and

obtaining security clearance to complete the requests;

monitoring shared memory comprising:

providing a call to reserve a memory page for a requesting process;

filtering the reserve call according to whether the page can be shared;

providing a call to commit the memory page for the requesting process or for a subsequent requesting process; and

filtering the commit call according to whether the page can be shared and whether the process can be secured;

providing a vault system for segregating vault data from other system data;

providing a file system security driver which intercepts file system calls, and for each specific one of said intercepted file system calls, determines whether said specific one of said intercepted file system calls is from a process accessing said vault data, and, if said specific

one of said intercepted file system calls is from a process accessing said vault data, permitting the file system call to create or modify data only within said vault system;

monitoring a system clock of a computer to prevent unauthorized access to data comprising:

initializing a clock monitor comprising the steps reading a first time value from the system clock;

determining whether a permissions database having one or more clock-related permission field each field comprising one or more clock-related permissions, and a stored time value field comprising a stored time value, is initialized on the computer system;

if the permissions database is initialized, comparing the first time value to the stored time value and, if the first time value is later than the stored time value, storing the first time value in the stored time value field;

if the first time value is earlier than the stored time value, disabling the one or more clock-related permissions, whereby disabling the clock-related permissions prevents access to the data;

if the permissions database is not initialized, storing the first time value in the stored time value field;

detecting an I/O request to said first device driver;

determining whether said first device driver is functionally uppermost in the layered plurality of device drivers;

if said first device driver is functionally uppermost in the layered plurality of device drivers, performing the I/O request in said first device driver; and

if said first device driver is not functionally uppermost in the layered plurality of device drivers, denying the I/O request in said first device driver, and allowing the I/O request to be performed by a next lower-level device driver in the layered plurality of device drivers;

detecting a port request for use of a port sent by a process;

determining the identity of said requesting process;

determining if said process should be permitted to access said port; allowing said port request to be fulfilled if said process should be permitted to access said port and denying said port request if said process should not be permitted to access said port;

detecting a port request for use of a port sent by a process;

if said port request is an open port request, determining the identity of said requesting process;

if said port request is an open port request, determining if said process should be permitted to open said port;

if said port request is an open port request, allowing said open port request to be fulfilled and tracking said open port request if said process should be permitted to open said port and denying said port request if said process should not be permitted to open said port;

if said port request is a close port request, completing said port request;

logging the closing of said port;

detecting a network request for use of a network sent by a process;

determining the identity of said requesting process;

determining if said process should be permitted to access said network; and

allowing said network request to be fulfilled if said process should be permitted to access said network and denying said network request if said process should not be permitted to access said network;

detecting a file system request; completing said file system request;

receiving return information from said file system request;

determining whether said file system request is for a tag file associated with a secured file; and

if so, modifying said return information to reflect a file attribute of the secured file.